# POOMA Class Development Process
Version 1.0
December 28, 1995

1. Class Description
   In this stage of the process, a written description of the class is developed. This document focuses on the class functionality, implementation strategies, classes within the FrameWork which would interface with the new class, and helper classes which would be required to enable the new class.
2. Description Review
   The Class description should be reviewed with the full team to insure proper class functionality and interaction with other classes within the FrameWork.
3. Interface Design
   This step entails producing a fully commented .h file for the proposed class and all required helper classes. Any changes to existent classes should be fully commented
4. Interface Review
   Before proceeding with code implementation, the new .h files and changes to existent .h files are reviewed by other team members.
5. Implementation
   Upon addressing concerns expressed at the interface review, the class is cast into source code.
6. Style Check
   At this stage, the pilot and co-pilot should ensure that all source code adheres to the POOMA style guide.
7. Debugging, Purification, and Test Suite Generation
   Before proceeding to the portability check, the pilot and co-pilot should debug as thoroughly as possible building a comprehensive test suite for the new class. All code should be run through Purify to ensure proper memory behavior.
8. Portability Check
   As a last check before code review, the new class should be exercised on all target compilers and architectures for the POOMA FrameWork. The serial C++ compilers include SGI Delta, SunOS, Photon, HP, and IBM R6K. The target architectures include the IBM SP2, the SGI PowerChallenge, and the Cray T3D.
9. Code Review
   At this point the source code is provided to other members of the team for reading, inspection, and walkthrough. By ensuring that other folks are able to follow the logic of the class implementation, one is ensured of improved software quality.
10. User's Guide
    While the code is under review, the pilot and co-pilot are responsible for generating a section of the POOMA user's guide describing the functionality of the new class
11. Hostile Testing and Test Suite Generation
    Equipped with the User's guide, the public interface, and the makeable source of the new class, the Red team now sets to the task of breaking the new class. All anomalous behavior should be reported as concise code segments which reproduce the bugs. As testing ensues, the Red team is responsible for building upon the test suite already in place so as to ensure a full battery of testing capabilities at the end of this process.
12. Bug Fixes

Here the Pilot and Co-pilot iterate with the Red Team to fix all reported problems.

## 13. TestCode verification

The new class should be integrated into the latest version of the FrameWork. However, this merged code should not be checked back in until all of the codes in the current test suite have been passed.

## 14. FrameWork update

Upon completion of the above processes, the source code should be checked into the latest CVS repository for the FrameWork, the Class description should be integrated into the POOMA Manual, the Class users Guide should be integrated into the full User’s Guide, and the Pilot and Co-Pilot should go out and celebrate.

## 15. POOMA Manual update

Upon completion of the CVS checking, the Class description should be integrated into the POOMA Manual.

## 16. User’s Guide update

Upon completion of the POOMA Manual update,  the Class users Guide should be integrated into the full User’s Guide.

## 17. Web-site update

Upon completion of the above processes, the latest POOMA Manual and User’s guide should be generated in .ps and .pdf formats and integrated into the POOMA web-site at http://www.acl.lanl.gov/PoomaFramework

I. Furthermore, the .public interface to the new class should be placed into a new link to the web site in addition to a class description.